IMPROVED PROPOSAL DISTRIBUTION WITH GRADIENT MEASURES FOR TRACKING

Paul Brasnett, Lyudmila Mihaylova, David Bull* and Nishan Canagarajah**

Dept. of Electrical and Electronic Engineering
University of Bristol
paul.brasnett@bristol.ac.uk, mila.mihaylova@bristol.ac.uk

ABSTRACT

Particle filters have become a useful tool for the task of object tracking due to their applicability to a wide range of situations. To be able to obtain an accurate estimate from a particle filter a large number of particles is usually necessary. A crucial step in the design of a particle filter is the choice of the proposal distribution. A common choice for the proposal distribution is to use the transition distribution which models the dynamics of the system but takes no account of the current measurements. We present a particle filter for tracking rigid objects in video sequences that makes use of image gradients in the current frame to improve the proposal distribution. The gradient information is efficiently incorporated in the filter to minimise the computational cost. Results from synthetic and natural sequences show that the gradient information improves the accuracy and reduces the number of particles required.

1. INTRODUCTION

Tracking objects in video is an important task due to its applications in diverse areas such as augmented reality, medical applications and surveillance. The general aim of tracking is to keep track of the pose and location of one or more objects through a sequence of frames.

Particle filtering [1, 2, 3, 4] is a powerful approach for tracking because it makes no assumptions of Gaussian noises and it is able to cope with highly non-linear models describing the image features and system dynamics. Challenges arise when accurate state representations are required in (near) real-time. To obtain results that are accurate a very good model for the proposal distribution is needed. Often the transition distribution is used to model the proposal distribution, this choice does not usually model the proposal distribution accurately so a large number of particles are required. As one would expect increasing the number of particles increases the complexity of the algorithm. This is particularly true for tracking in video sequences because the cost of evaluating the likelihood tends to be high.

*This work has been conducted with support from the UK MOD Data and Information Fusion Defence Technology Centre under project DIF DTC 2.2.

There are a number of variants of the particle filter that attempt to address the problem of the proposal distribution. Some of them rely on additional strategies for the proposal distribution such as Monte Carlo Markov chains or the use of gradient information in order to move particles toward more likely regions. The idea of using gradient information in the proposal distribution has previously been applied to the area of wireless communications [5]. An additional *MOVE* step is introduced before the sampling step. The gradient information to guide the move to regions of higher likelihood is calculated from the likelihood model.

In [6] the generation of particles is controlled by a *momentum* term. Particles with a momentum below a threshold are propagated through a deterministic, gradient-descent search the remaining particles are propagated by sampling from the transition density function. An alternative method that moves particles toward regions of higher likelihood is the Kernel Particle Filter [7]. In this approach the mean shift tracker [8] is embedded in a particle filter. Following resampling the mean shift iteratively estimates the local likelihood density gradient and moves the particles toward stationary points, which include the modes. The result is that particles are focused around stationary points in the likelihood density.

The aim of the work here is in a similar vein to the works mentioned above, in that gradient information is used to shift the particles. In the present paper an error function is defined and optimised in an efficient gradient-descent method based on the image gradient information available in the frame. In [5] a Levenburg-Marquardt optimisation approach is used whilst here a Newton-Raphson approach to gradient descent allows the development of an efficient implementation. This approach works for a range of linear and non-linear motions, including common motions such as the translation and affine models. The benefit of embedding this in a particle filter framework is the ability to maintain multiple hypothesis, something not possible in a purely deterministic framework.

An introduction to particle filtering is provided in Section 2. Details of the colour histogram based likelihood

model are included in Section 3. A general description of the gradient descent is given in Section 4, from this an efficient gradient measurement for video sequences is developed. The gradient information is incorporated into the particle filtering framework along with implementation details in Section 5. Results are presented in Section 6 and conclusions are given in Section 7.

2. PARTICLE FILTERING

Given a system transition function $f_t : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$

$$\boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{w}_t), \tag{1}$$

the system state vector $\boldsymbol{x}_t \in \mathbb{R}^n$ is estimated at time t where $\boldsymbol{w}_t \in \mathbb{R}^m$ is a zero mean, white noise sequence independent of past and current states with a known probability density function (PDF).

Table 1: Generic Particle Filter

1. **Initialisation**
   For $n = 1, \ldots, N$ set $w_0^n = \frac{1}{N}$.
2. **Importance Sampling**
   For $n = 1, \ldots, N$
   - Sample $\boldsymbol{x}_{t+1}^n \sim q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^n, \boldsymbol{y}_{t+1})$
   - Evaluate the weights
   $$w_{t+1}^n = w_t^n \frac{p(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^n)p(\boldsymbol{x}_{t+1}^n|\boldsymbol{x}_t^n)}{q(\boldsymbol{x}_{t+1}^n|\boldsymbol{x}_t^n, \boldsymbol{y}_{t+1})} \tag{2}$$
   - Normalise the weights, $\tilde{w}_{t+1}^n = \frac{w_{t+1}^n}{\sum_{m=1}^N w_{t+1}^m}$

   Evaluate $\hat{N}_{eff} = \frac{1}{\sum_{n=1}^N (\tilde{w}_{t+1}^n)^2}$
3. **Output**
   Estimate the current state
   $$\hat{\boldsymbol{x}}_{t+1} = \sum_{n=1}^N \tilde{w}_{t+1}^n \boldsymbol{x}_{t+1}^n. \tag{3}$$
4. **Resampling**
   If $\hat{N}_{eff} \leq N_{thres}$
   - For $n = 1, \ldots, N$, resample with replacement N particles x_{t+1}^i according to their weights, where N_{thres} is a given threshold value.

Measurements $\boldsymbol{y}_t \in \mathbb{R}^p$ are related to the state vector via the observation equation

$$\boldsymbol{y}_t = h_t(\boldsymbol{x}_t, \boldsymbol{v}_t), \tag{4}$$

where $h_t : \mathbb{R}^n \times \mathbb{R}^r \rightarrow \mathbb{R}^p$ is the measurement function and $\boldsymbol{v}_t \in \mathbb{R}^r$ is a different zero mean, white noise sequence with known PDF, independent of past and present states of the system noise. The Bayesian interpretation of the tracking problem is to recursively calculate a degree of belief in the state $\boldsymbol{x}_t$ at time t given the measurements $\boldsymbol{y}_{1:t} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$. This is represented by the posterior PDF $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$.

The posterior PDF $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ of the state $\boldsymbol{x}_t$ is approximated by the particle filter given a measurement $\boldsymbol{y}_{1:t}$ and a set of particles $\boldsymbol{x}_t^n$ each with a corresponding weight w_t^n

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \approx \sum_{n=1}^N w_t^n \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^n), \tag{5}$$

where $\delta(.)$ is the Kronecker delta function. Each one of the particles $\boldsymbol{x}_{t+1}^n$ is drawn from the proposal distribution $q(\boldsymbol{x}_{t+1}^n|\boldsymbol{x}_{0:t}^n, \boldsymbol{y}_{1:t+1})$ and assigned a weight w_{t+1}^n calculated recursively at each time step by evaluating the transition density $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$ the likelihood $p(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1})$ and the evidence $q(\boldsymbol{x}_{t+1}^n|\boldsymbol{x}_t^n, \boldsymbol{y}_{t+1})$. The generic particle filter is given in Table 1.

Note that the resampling stage of Table 1 is necessary to limit the effects of degeneracy [4, 9], the case when only one particle has significant weight.

In the design of a particle filter it is critical to choose a suitable importance density $q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{0:t}^n, \boldsymbol{y}_{0:t+1})$. A common choice for the importance density $q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^n, \boldsymbol{y}_{t+1})$ is to use the transition density

$$q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^n, \boldsymbol{y}_{t+1}) = p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^n). \tag{6}$$

This choice only takes into account the system dynamics, no account is taken of the measurements. The transition prior is chosen because it leads to a straightforward implementation.

3. LIKELIHOOD MODEL

Weighted colour histogram cues extracted from the frame are used as the result of the measurement function. The weighted histogram $H_{i,\boldsymbol{x}}$ for bin i and state $\boldsymbol{x}$ is given by

$$H_{i,\boldsymbol{x}} = C_H \sum_{\boldsymbol{r} \in \mathcal{S}_{\boldsymbol{x}}} k_N \left(\left\| \frac{\bar{\boldsymbol{r}} - \boldsymbol{r}}{a} \right\|^2 \right) \delta_i(b_{\boldsymbol{r}}), \quad i = 1 \ldots B, \tag{7}$$

where $\bar{\boldsymbol{r}} = (\bar{x}, \bar{y})$ is the location of the center pixel, C_H is a normalisation constant such that $\sum_{i=1}^B H_{i,\boldsymbol{x}} = 1$, a is the size of the kernel, $b_{\boldsymbol{r}} \in \{1, \ldots, B\}$ denotes the histogram bins, $\delta_i(.)$ is the Kronecker delta function at i and $\mathcal{S}_{\boldsymbol{x}}$ is the set of pixel locations $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_R\}$ defined by the state $\boldsymbol{x}$ and the model g (see Section 4.2). The Gaussian kernel,

k_N, is used to weight pixels in the center of the region more highly than pixels at the edge of the region

$$k_N(\boldsymbol{r}) = (2\pi)^{-1/2} e^{-\frac{1}{2}\boldsymbol{r}}. \quad (8)$$

The Bhattacharyya coefficient ρ determines the distance between two histograms

$$\rho(\boldsymbol{H}_{\text{ref}}, \boldsymbol{H}_{\text{tar}}) = \sum_{i=1}^{B} \sqrt{H_{i,\text{ref}} H_{i,\text{tar}}}. \quad (9)$$

where two normalised histograms $\hat{\boldsymbol{H}}_{\text{tar}}$ and $\hat{\boldsymbol{H}}_{\text{ref}}$ represent a *target region* defined in the current frame and a *reference region* in the frame at t_0. The Bhattacharyya distance [8]

$$d(\boldsymbol{H}_{\text{tar}}, \boldsymbol{H}_{\text{ref}}) = \sqrt{1 - \rho(\boldsymbol{H}_{\text{tar}}, \boldsymbol{H}_{\text{ref}})}, \quad (10)$$

is a measure of the similarity between these two distributions. The larger the measure $\rho(\boldsymbol{H}_{\text{ref}}, \boldsymbol{H}_{\text{tar}})$ is, the more similar the distributions are. Conversely, for the distance d, the smaller the value the more similar the distributions (histograms) are. For two identical normalised histograms we obtain $d = 0$ ($\rho = 1$) indicating a perfect match.

Based on this distance the *likelihood function* over red, green, blue (R, G, B) colour space can be defined by [10]

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t^n) \propto exp\left(-\sum_{c\in\{R,G,B\}} \frac{d^2(\boldsymbol{H}_{\text{tar}}^c, \boldsymbol{H}_{\text{ref}}^c)}{2\sigma_c^2}\right), \quad (11)$$

for the n-th particle $\boldsymbol{x}_t^n$. The standard deviation σ specifies the Gaussian noise in the measurements. Note that small Bhattacharyya distances correspond to large weights in the particle filter.

4. GRADIENT INFORMATION

The aim of the gradient descent is to minimise an objective function, O, with respect to the state vector $\boldsymbol{x}$,

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} O(\boldsymbol{x}). \quad (12)$$

If $\boldsymbol{I}_t(\boldsymbol{x}_t) = [I(\boldsymbol{r}_1, t), \ldots, I(\boldsymbol{r}_R, t)]'$ is the vector of R pixel intensities from an image region $\mathcal{S}_{\boldsymbol{x}_t}$, corresponding to state $\boldsymbol{x}_t$ at time t. Furthermore the locations $\boldsymbol{r} = [x, y]'$ in $\mathcal{S}_{\boldsymbol{x}_t}$ are determined by the model g. It is assumed that $\boldsymbol{g}$ is differentiable with respect to both $\boldsymbol{r}$ and $\boldsymbol{x}$.

The objective function can specifically be defined as the following least squares function [11]

$$O(\boldsymbol{x}) = \sum_{i\in\boldsymbol{I}} (\boldsymbol{I}_t(\boldsymbol{x}_t) - \boldsymbol{I}_{t_0}(\boldsymbol{x}_0))^2, \quad (13)$$

where $\boldsymbol{x}_0$ is the initial state at time t_0. Alternatively the objective function can be expressed as

$$O(\boldsymbol{x}_t) = \|\boldsymbol{I}_t(\boldsymbol{x}_t) - \boldsymbol{I}_{t_0}(\boldsymbol{x}_0)\|^2. \quad (14)$$

Reposing the problem in terms of iteratively determining the offset $\delta\boldsymbol{x}$ such that $\hat{\boldsymbol{x}_t} = \boldsymbol{x}_t + \delta\boldsymbol{x}$ then (14) becomes

$$O(\delta\boldsymbol{x}) = \|\boldsymbol{I}_t(\boldsymbol{x}_t + \delta\boldsymbol{x}) - \boldsymbol{I}_{t_0}(\boldsymbol{x}_0)\|^2. \quad (15)$$

If we assume that $\delta\boldsymbol{x}$ is small then we can apply continuous optimisation procedures to a linearised version of the problem. The problem can be linearised by performing a Taylor series expansion of $\boldsymbol{I}_t(\boldsymbol{x}_t + \delta\boldsymbol{x})$ about $\boldsymbol{x}_t$

$$\boldsymbol{I}_t(\boldsymbol{x}_t + \delta\boldsymbol{x}) \approx \boldsymbol{I}_t(\boldsymbol{x}_t) + \delta\boldsymbol{x}\boldsymbol{M}_t(\boldsymbol{x}_t) + \text{H.O.T.}, \quad (16)$$

where H.O.T. refers to higher order terms of the Taylor series expansion and $\boldsymbol{M}_t$ is the Jacobian matrix of $\boldsymbol{I}_t$ with respect to $\boldsymbol{x}_t$. Making the substitution of (16) into (15) gives

$$O(\delta\boldsymbol{x}) \approx \|\boldsymbol{I}_t(\boldsymbol{x}_t) + \delta\boldsymbol{x}\boldsymbol{M}_t(\boldsymbol{x}_t) - \boldsymbol{I}_{t_0}(\boldsymbol{x}_0)\|^2. \quad (17)$$

Solving for $\frac{\partial O}{\partial(\delta\boldsymbol{x})} = 0$ and rearranging gives

$$\delta\boldsymbol{x} = -(\boldsymbol{M}_t'\boldsymbol{M}_t)^{-1}\boldsymbol{M}_t'[\boldsymbol{I}_t(\boldsymbol{x}_t) - \boldsymbol{I}_{t_0}(\boldsymbol{x}_0)], \quad (18)$$

and from this $\hat{\boldsymbol{x}_t}$ can be defined as

$$\hat{\boldsymbol{x}_t} = \boldsymbol{x}_t - (\boldsymbol{M}_t'\boldsymbol{M}_t)^{-1}\boldsymbol{M}_t'\boldsymbol{e}_t, \quad (19)$$

where

$$\boldsymbol{e}_t = \boldsymbol{I}_t(\boldsymbol{x}_t) - \boldsymbol{I}_{t_0}(\boldsymbol{x_0}). \quad (20)$$

4.1. Efficient Algorithm

Evaluating (19) requires the estimation of the gradient of each target region in every frame. To allow efficient online implementation it can be shown that $\boldsymbol{M}_t$ can be decomposed into a time-varying component $\boldsymbol{\Sigma}_t$ and a constant $\boldsymbol{M}_0$, which can be determined *off-line*. The efficiency comes from removing the need to recalculate the Jacobian $\boldsymbol{M}_t$ at every iteration. The decomposition of $\boldsymbol{M}_t$ is

$$\boldsymbol{M}_t(\boldsymbol{x}_t) = \begin{bmatrix} \nabla_{\boldsymbol{r}} I(\boldsymbol{r}_1, t_0)'\boldsymbol{\Gamma}(\boldsymbol{r}_1) \\ \nabla_{\boldsymbol{r}} I(\boldsymbol{r}_2, t_0)'\boldsymbol{\Gamma}(\boldsymbol{r}_2) \\ \vdots \\ \nabla_{\boldsymbol{r}} I(\boldsymbol{r}_R, t_0)'\boldsymbol{\Gamma}(\boldsymbol{r}_R) \end{bmatrix} \boldsymbol{\Sigma}_t(\boldsymbol{x}) = \boldsymbol{M}_0\boldsymbol{\Sigma}_t(\boldsymbol{x}), \quad (21)$$

where $\nabla_{\boldsymbol{r}} I(\boldsymbol{r}_\ell, t_0)'$, $\ell = 1, \ldots, R$ denotes the gradient, with respect to the components of $\boldsymbol{r}$, of pixel $\boldsymbol{r}_\ell$ at time t_0, $\boldsymbol{\Sigma}_t(\boldsymbol{x}_t)$ is dependent upon the motion model used and $\boldsymbol{\Gamma}(\boldsymbol{r})$ depends on both the motion model g used and the pixel location $\boldsymbol{r} = [x, y]'$. Examples of g are given for the translation model (Section 4.2) and the affine model (Section 4.3).

If $\boldsymbol{\Sigma}_t$ is invertible then the state can be moved toward the minimum of the error vector $\boldsymbol{e}_t$ by

$$\hat{\boldsymbol{x}_t} = \boldsymbol{x}_t - (\boldsymbol{\Sigma}_t^{-1})'\boldsymbol{\Lambda}\boldsymbol{e}_t \quad (22)$$

where $\boldsymbol{\Lambda} = (\boldsymbol{M}_0'\boldsymbol{M}_0)^{-1}\boldsymbol{M}_0$ and is computed during an initialisation stage based on the model used. Hence, equation (19) is replaced by the more computationally efficient (22).

4.2. Translation Motion Model

The motion model described here and in section 4.3 defines how the pixel locations are related to the state. This first model is for the case when the object performs a translation motion

$$f(\boldsymbol{r}, \boldsymbol{x}_t) = \boldsymbol{r} + \boldsymbol{x}_t, \quad \boldsymbol{x}_t = [u_t, v_t]' \tag{23}$$

and for this model $\boldsymbol{M}_0 = [\boldsymbol{I}_x(t_0)|\boldsymbol{I}_y(t_0)]$ and $\boldsymbol{\Sigma}$ is the 2×2 identity matrix. Remembering that $\boldsymbol{\Lambda} = (\boldsymbol{M}_0^T \boldsymbol{M}_0)^{-1} \boldsymbol{M}_0$ the updated state, $\hat{\boldsymbol{x}}_t$, at time t is given by

$$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\Lambda} \boldsymbol{e}_t. \tag{24}$$

4.3. Affine Motion Model

If the object to be tracked is a planar object a more suitable model to capture the transformation is given by the six component affine transform. The motion model and current state of the object $\boldsymbol{x}_t$ at time t can be described by

$$f(\boldsymbol{r}, \boldsymbol{x}_t) = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \boldsymbol{r} + \begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{A}\boldsymbol{r} + \boldsymbol{u}. \tag{25}$$

The state vector is $\boldsymbol{x}_t = [u_t, v_t, a_t, b_t, c_t, d_t]'$. Using the affine motion model gives

$$\boldsymbol{\Gamma}(\boldsymbol{p}) = \begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & x & 0 & y \end{bmatrix}, \tag{26}$$

and

$$\boldsymbol{\Sigma}_t(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^{-}1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}^{-1} \end{bmatrix}. \tag{27}$$

The updated state, $\hat{\boldsymbol{x}}_t$, at time t is given by

$$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\Sigma}_t' \boldsymbol{\Lambda} \boldsymbol{e}_t. \tag{28}$$

It is possible to use other models including some non-linear models. Not all of them are suitable because of the separability property needed to factorise $\boldsymbol{M}$.

5. IMPLEMENTATION

Table 2 presents a particle filter that takes into account the gradient information in the way described in Section 4.

Table 2: Particle Filter with Gradient Step

1. **Initialisation**
   For $n = 1, \ldots, N$ set $w_0^n = \frac{1}{N}$.
   Calculate $\boldsymbol{M}_0 = [\boldsymbol{I}_x(t_0)|\boldsymbol{I}_y(t_0)]$ for the target region and then evaluate
   $$\boldsymbol{\Lambda} = (\boldsymbol{M}_0^T \boldsymbol{M}_0)^{-1} \boldsymbol{M}_0 \tag{29}$$
2. **Importance Sampling**
   For $n = 1, \ldots, N$,
   - Sample $\boldsymbol{x}_{t+1|t}^n \sim q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{t|t}^n)$
   - For $j = 1, \ldots, J$ (with J iterations)
     - Gradient step
       $\boldsymbol{x}_{t+1|t}^n = \boldsymbol{x}_{t+1|t}^n - \boldsymbol{\Omega}\boldsymbol{\Sigma}_t' \boldsymbol{\Lambda} \boldsymbol{e}_{t+1}^n$
   - Evaluate the weights
     $$w_{t+1}^n = w_t^n p(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^n) \frac{p(\boldsymbol{x}_{t+1}^i|\boldsymbol{x}_{t|t}^i)}{p(\boldsymbol{x}_{t+1}^i|\boldsymbol{x}_{t+1|t}^i)} \tag{30}$$
   - Normalise the weights, $\tilde{w}_{t+1}^n = \frac{w_{t+1}^n}{\sum_{m=1}^N w_{t+1}^m}$

   Evaluate $\hat{N}_{eff} = \frac{1}{\sum_{n=1}^N (\tilde{w}_{t+1}^n)^2}$
3. **Output**
   Estimate the current state
   $$\bar{\boldsymbol{x}}_{t+1} = \sum_{n=1}^{N} \tilde{w}_{t+1}^n \boldsymbol{x}_{t+1}^n. \tag{31}$$
4. **Resampling**
   If $\hat{N}_{eff} \leq N_{thres}$
   - For $n = 1, \ldots, N$, resample with replacement N particles $\boldsymbol{x}_{t+1}^n$ according to their weights

For the purpose of tracking an object in video we initially choose a region which defines the object. The shape of this region is fixed *a priori* and in our case it is a rectangular box characterised by the state vector $\boldsymbol{x} = (x, \dot{x}, y, \dot{y})'$, with x and y denoting the pixel location of the top-left corner of the rectangle, with velocities $\dot{x}$ and $\dot{y}$. Note that the dimensions of the rectangle are fixed through the sequence.

The transition distribution $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$ used for this work is a constant velocity dynamic model [12]

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}\boldsymbol{x}_k + \boldsymbol{w}_k, \quad \boldsymbol{w}_k \backsim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}) \tag{32}$$

$$\boldsymbol{F} = \begin{pmatrix} \tilde{\boldsymbol{F}} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{F}} \end{pmatrix}, \qquad \tilde{\boldsymbol{F}} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix},$$
$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_x & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_y \end{pmatrix}, \qquad \boldsymbol{\Gamma} = \begin{pmatrix} \frac{1}{2}T^2 \\ T \end{pmatrix},$$

with the state vector $\boldsymbol{x} = (x, \dot{x}, y, \dot{y})'$, the system noise $\boldsymbol{w} = (\boldsymbol{w}'_x, \boldsymbol{w}'_y)' = (\boldsymbol{\Gamma}' v_{x,k}, \boldsymbol{\Gamma}' v_{y,k})'$, $v_{x,k} \backsim \mathcal{N}(0, \sigma_x)$, $v_{y,k} \backsim \mathcal{N}(0, \sigma_y)$ being scalar valued zero mean white sequences with standard deviations σ_x and σ_y respectively and T is the sampling interval.

The covariance matrices of the noise respectively in x and y coordinates multiplied by the gain, are

$$\boldsymbol{Q}_x = \boldsymbol{\Gamma}\sigma_x^2\boldsymbol{\Gamma} = \begin{pmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 \\ \frac{1}{2}T^3 & T^2 \end{pmatrix} \sigma_x^2.$$

The covariance $\boldsymbol{Q}_y$ can be calculated in a similar way. Suitable values for σ_x and σ_y are ([12], p. 273) in the range $[\frac{1}{2}a_m, a_m]$, with a_m being the maximum acceleration.

An implementation issue in combining the constant velocity model with the gradient descent is that the gradient descent only updates the x and y coordinates of the state and appropriate account needs to be taken to update the velocities. This is done through the use of the following matrix

$$\boldsymbol{\Omega} = \begin{bmatrix} 1 & \frac{1}{T} & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{T} \end{bmatrix}', \tag{33}$$

where T is the sampling period and in our implementation $T = 1$.

6. RESULTS

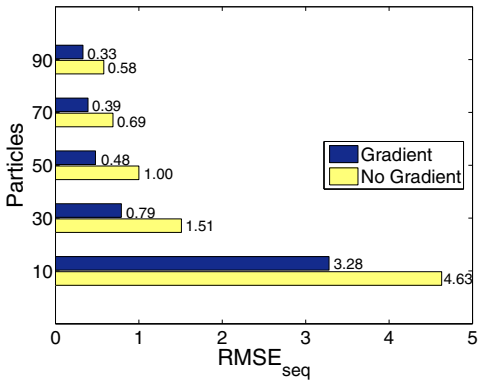
The results presented here are from experiments carried out on rigid objects in a natural sequences (Fig. 1) and a synthetic sequence (Fig. 2). The object in the synthetic sequence is quite textured, in the artificial sequence it contains more homogeneous regions. The target regions are initialised by providing the coordinates of the target region in the first frame. The state $\boldsymbol{x}_t = (\hat{x}, \hat{y})$ represents an estimate of the true coordinates (x, y), therefore the root mean square $RMSE$ is

$$RMSE = \sqrt{(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2}. \tag{34}$$

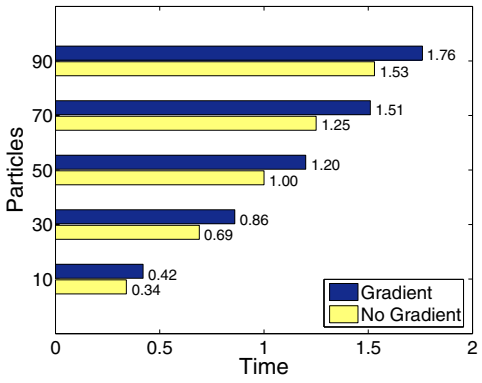
and for a sequence of F frames it is

$$RMSE_{\text{seq}} = \sqrt{\frac{1}{F}\sum_{t=1}^{F}(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2}. \tag{35}$$

A comparison of the relative performance of the two algorithms over the natural sequence is given in Fig. 1. It can be seen that for any number of particles more accurate results are obtained by the particle filter with a gradient step.



(a) Relative Performance



(b) Relative Complexity

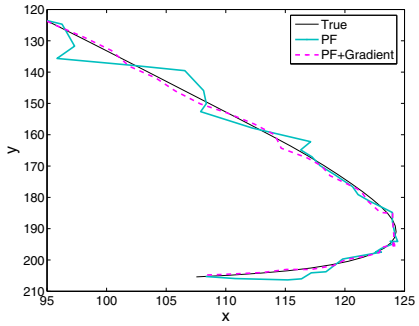
Fig. 1. Comparison of the generic particle filter and particle filter with gradient step. All of the results are presented as relative to the generic particle filter with 50 particles (a) Relative RMSE of the state estimated by the algorithm. (b) Relative processing time for the generic particle filter and the particle filter with gradient step. The results are for a sequence of 60 frames and are averaged over 100 runs.

It can also be seen that for comparable complexity the gradient particle filter outperforms the generic particle filter. Tracking results of the two algorithms on the natural sequence are shown in Fig. 3.

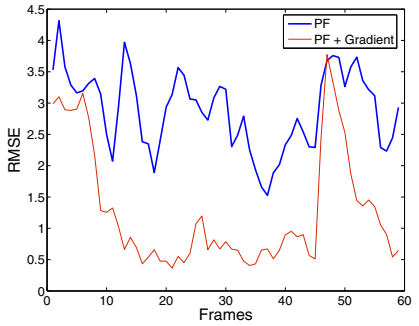
Results from a synthetic sequence are shown in Fig. 2. The estimated path from the particle filter is clearly smoother and more accurate when the gradient information is used. This can be clearly seen in Fig. 2a by the jittering in the particle filter path that is not present in when the gradient step is used. The $RMSE$ can be clearly seen to be lower when the gradient information is used in Fig. 2b.

7. CONCLUSIONS

We have presented a method of improving the proposal distribution in the particle filter by taking into account the gradient information available in a frame. The inclusion of information from the current frame in the proposal distri-



(a) Tracking Results



(b) RMSE

Fig. 2. Comparison of the generic particle filter and particle filter with gradient step applied to an synthetic sequence. (a) A section of the true path of the object compared to the estimates from one run of the particle filter and the particle filter with the gradient step. (b) The RMSE of the particle filter and the particle filter with the gradient step, the results are the mean of 100 run.

bution moves the particles to the high-likelihood regions which results in improved performance. An efficient implementation of the gradient step particle filter is given. Results show that there are two main improvements over a generic particle filter i) a significant increase in the accuracy (lower $RMSE$) and ii) since fewer particles are needed to represent the posterior there is a reduction in computational complexity. Future work will extend the method to include changes to the object appearance.

8. REFERENCES

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. on Signal Proc.*, vol. 50, no. 2, pp. 174–188, 2002.

[2] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[3] N. Gordon, D. Salmond, and A. Smith, "A novel approach to nonlinear / non-Gaussian Bayesian state estimation," *IEE Proc. on Radar and Signal Processing*, vol. 40, pp. 107–113, 1993.

[4] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[5] S. Haykin, K. Huber, and Z. Chen, "Bayesian sequential state estimation for MIMO wireless communications," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 439–454, March 2004.

[6] J. Sullivan and J. Rittscher, "Guiding random particles by deterministic search," in *IEEE Int. Conf. on Computer Vision*, July 2001, vol. 1, pp. 323–330.

[7] C. Chang and R. Ansari, "Kernel particle filter for visual tracking," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 242–245, March 2005.

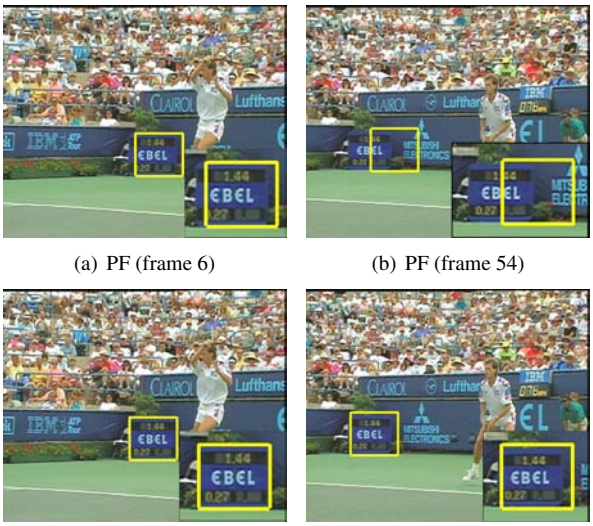
[8] D. Comaneciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[9] A. Kong, J. Liu, and W. Wong, "Sequential imputations and Bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, pp. 278–288, 1994.

[10] P. Pérez, J. Vermaak, and A. Blake, "Data fusion for tracking with particles," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 495–513, March 2004.

[11] G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. on PAMI*, vol. 20, no. 10, pp. 1025–1039, 1998.

[12] Y. Bar-Shalom and X.R. Li, *Estimation and Tracking: Principles, Techniques and Software*, Artech House, 1993.



(a) PF (frame 6) (b) PF (frame 54)

(c) PF with Gradient (frame 6) (d) PF with Gradient (frame 54)

Fig. 3. Frames (6 and 54) from a sequence tracking the scoreboard with the two filters. It can be seen that the particle filter with a gradient step results in more accurate tracking than the generic particle filter.